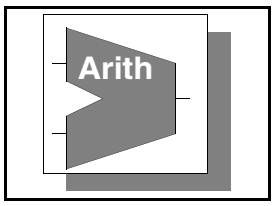
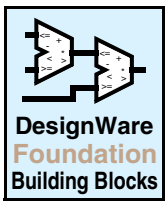



# DW_norm

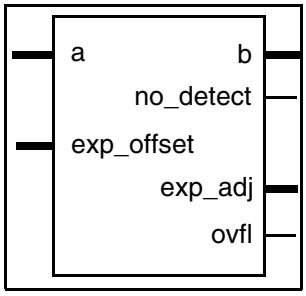
## Normalization for Fractional Input

Version, STAR, and myDesignWare Subscriptions: IP Directory

## Features and Benefits

- Parameterized word lengths
- Parameterized search window
- Indication of exceptional cases

## Revision History




## Description

DW_norm is a general-purpose normalization module for positive fractional input. The fixed-point input format is a = ($a_0.a_1\ a_2\ a_3\ a_4\ a_5 \ldots a_{a\_width-1}$), where $a_i$ represents a bit. Input a has one integer bit and *a_width* - 1 fractional bits. The normalization process consists in shifting the input vector a to the left until the output bit-vector has a 1 in the MS bit position, or the vector was shifted by the maximum number of bits in the search window.

The number of bit positions shifted to the left during normalization (*n*) is passed to the value of *exp_adj*. This output corresponds to (*exp_offset* + *n*) when parameter *exp_ctr* = 0 or (*exp_offset* - *n*) when *exp_ctr* = 1, where *n* = *max*(*srch_wind* - 1, *number_of_MS_zeros*).

**Table 1-1 Pin Description**

| Pin Name | Width | Direction | Function |
|---|---|---|---|
| a | *a_width* bits | Input | Input data |
| exp_offset | *exp_width* bits | Input | Offset value for the exponent |
| no_detect | 1 bit | Output | Result of search for the leading bit with value 1 in the search window<br>■ 0: Bit found<br>■ 1: Bit not found |
| ovfl | 1 bit | Output | Value provided at output *exp_adj* is negative or incorrect |
| b | *a_width* bits | Output | Normalized output data |
| exp_adj | *exp_width* bits | Output | *exp_offset* combined with the number of bit positions the input *a* was shifted to the left (*n*)<br>■ *exp_ctr* = 0 → *exp_offset* + *n*<br>■ *exp_ctr* = 1 → *exp_offset* - *n* |

**Table 1-2 Parameter Description**

| Parameter | Values | Description |
|---|---|---|
| a_width | ≥ 2<br>Default: 8 | Word length of `a` and `b` |
| srch_wind | 2 to *a_width*<br>Default: 8 | Search window for the leading 1 bit (from bit position 0 to *a_width* - 1) |
| exp_width[a] | ≥ ceil(log2(*srch_wind*))<br>Default: 4 | Word length of *exp_offset* and *exp_adj* |
| exp_ctr | 0 or 1<br>Default: 0 | Control over *exp_adj* computation |

a. The lower bound of parameter *exp_width* was modified starting on G-2012.06-SP1; the previous lower bound was 1.

**Table 1-3 Synthesis Implementations**

| Implementation Name | Function | License Feature Required |
|---|---|---|
| rtl | Synthesis model for fast architecture | DesignWare |
| str | Synthesis model for reduced area | DesignWare |

**Table 1-4 Simulation Models**

| Model | Function |
|---|---|
| DW01.DW_NORM_CFG_SIM | Design unit name for VHDL simulation |
| dw/dw01/src/DW_norm_sim.vhd | VHDL simulation model source code |
| dw/sim_ver/DW_norm.v | Verilog simulation model source code |

The input *exp_offset* is provided by the user to indicate that a pre-shifting of the input operand was already done to adjust it to the required input format. For example, if the actual fixed-point value is (101.0011), the input for the DW_norm component may be *a* = (0.01010011) and *exp_offset* = (0100), for the parameters *a_width* = 9 and *exp_width* = 4. It shows the case of left zero padding on the bit vector.

The *srch_wind* parameter is used to simplify the normalization hardware. This parameter defines the number of MS bits that must contain a 1. When there is no leading 1 in the search window, the output is shifted to the left by *srch_wind* - 1 bit positions, and the *no_detect* output is set to 1.

After normalization, using the same numerical example as above, the outputs *b*, *no_detect*, and *exp_adj* are generated as *b* = (1.01001100), *no_detect* = 0, and *exp_adj* = (0010), when *srch_wind* > 2 and *exp_ctr* = 1. In this case, the number of bit positions shifted to the left was subtracted from the *exp_offset* value. If *srch_wind* = 2,

the leading 1 in the example provided above would not be found, and the output values would be *no_detect* = 1, *b* = (0.10100110), and *exp_adj* = (0011).

Output *ovfl* is 1 when the value provided at the *exp_adj* output is negative or incorrect (when the number of bits used in exp_adj is not enough to represent the number). In order to have a meaningful value in the *exp_adj* output, the parameter *exp_width* must be at least $\log_2$(*srch_wind*). Table 1-5 and Table 1-6 on page 4 show the behavior of this component.

**Table 1-5 Truth Table (*a_width* = 4, *exp_width* = 2, *srch_wind* = 4, *exp_ctr* = 0)**

| *a*(3:0) | *exp_offset* (1:0) | *no_detect* | *b*(0) | *b*(1) | *b*(2) | *b*(3) | *exp_adj* | *ovfl* |
|---|---|---|---|---|---|---|---|---|
| 0000 | 00 | 1 | 0 | 0 | 0 | 0 | 11 | 0 |
| 0001 | 00 | 0 | 1 | 0 | 0 | 0 | 11 | 0 |
| 001x | 00 | 0 | 1 | x | 0 | 0 | 10 | 0 |
| 01xx | 00 | 0 | 1 | x | x | 0 | 01 | 0 |
| 1xxx | 00 | 0 | 1 | x | x | x | 00 | 0 |
| 0000 | 01 | 1 | 0 | 0 | 0 | 0 | 00 | 1 |
| 0001 | 01 | 0 | 1 | 0 | 0 | 0 | 00 | 1 |
| 001x | 01 | 0 | 1 | x | 0 | 0 | 11 | 0 |
| 01xx | 01 | 0 | 1 | x | x | 0 | 10 | 0 |
| 1xxx | 01 | 0 | 1 | x | x | x | 01 | 0 |
| 0000 | 10 | 1 | 0 | 0 | 0 | 0 | 01 | 1 |
| 0001 | 10 | 0 | 1 | 0 | 0 | 0 | 01 | 1 |
| 001x | 10 | 0 | 1 | x | 0 | 0 | 00 | 1 |
| 01xx | 10 | 0 | 1 | x | x | 0 | 11 | 0 |
| 1xxx | 10 | 0 | 1 | x | x | x | 10 | 0 |
| 0000 | 11 | 1 | 0 | 0 | 0 | 0 | 10 | 1 |
| 0001 | 11 | 0 | 1 | 0 | 0 | 0 | 10 | 1 |
| 001x | 11 | 0 | 1 | x | 0 | 0 | 01 | 1 |
| 01xx | 11 | 0 | 1 | x | x | 0 | 00 | 1 |
| 1xxx | 11 | 0 | 1 | x | x | x | 11 | 0 |

**Table 1-6 Truth Table (*a_width*=4, *exp_width*=2, *srch_wind*=4, *exp_ctr*=1)**

| *a*(3:0) | *exp_offset* (1:0) | *no_detect* | *b*(0) | *b*(1) | *b*(2) | *b*(3) | *exp_adj* | *ovfl* |
|---|---|---|---|---|---|---|---|---|
| 0000 | 00 | 1 | 0 | 0 | 0 | 0 | 01 | 1 |
| 0001 | 00 | 0 | 1 | 0 | 0 | 0 | 01 | 1 |
| 001x | 00 | 0 | 1 | x | 0 | 0 | 10 | 1 |
| 01xx | 00 | 0 | 1 | x | x | 0 | 11 | 1 |
| 1xxx | 00 | 0 | 1 | x | x | x | 00 | 0 |
| 0000 | 01 | 1 | 0 | 0 | 0 | 0 | 10 | 1 |
| 0001 | 01 | 0 | 1 | 0 | 0 | 0 | 10 | 1 |
| 001x | 01 | 0 | 1 | x | 0 | 0 | 11 | 1 |
| 01xx | 01 | 0 | 1 | x | x | 0 | 00 | 0 |
| 1xxx | 01 | 0 | 1 | x | x | x | 01 | 0 |
| 0000 | 10 | 1 | 0 | 0 | 0 | 0 | 11 | 1 |
| 0001 | 10 | 0 | 1 | 0 | 0 | 0 | 11 | 1 |
| 001x | 10 | 0 | 1 | x | 0 | 0 | 00 | 0 |
| 01xx | 10 | 0 | 1 | x | x | 0 | 01 | 0 |
| 1xxx | 10 | 0 | 1 | x | x | x | 10 | 0 |
| 0000 | 11 | 1 | 0 | 0 | 0 | 0 | 00 | 0 |
| 0001 | 11 | 0 | 1 | 0 | 0 | 0 | 00 | 0 |
| 001x | 11 | 0 | 1 | x | 0 | 0 | 01 | 0 |
| 01xx | 11 | 0 | 1 | x | x | 0 | 10 | 0 |
| 1xxx | 11 | 0 | 1 | x | x | x | 11 | 0 |

## Related Topics

- Datapath– Arithmetic Overview
- DesignWare Building Block IP User Guide

## HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW_norm_inst is
      generic (
        inst_a_width : POSITIVE := 8;
        inst_srch_wind : POSITIVE := 8;
        inst_exp_width : POSITIVE := 4;
        inst_exp_ctr : INTEGER := 0
        );
      port (
        inst_a : in std_logic_vector(inst_a_width-1 downto 0);
        inst_exp_offset : in std_logic_vector(inst_exp_width-1 downto 0);
        no_detect_inst : out std_logic;
        ovfl_inst : out std_logic;
        b_inst : out std_logic_vector(inst_a_width-1 downto 0);
        exp_adj_inst : out std_logic_vector(inst_exp_width-1 downto 0)
        );
    end DW_norm_inst;


architecture inst of DW_norm_inst is

begin

    -- Instance of DW_norm
    U1 : DW_norm
    generic map ( a_width => inst_a_width, srch_wind => inst_srch_wind, exp_width =>
inst_exp_width, exp_ctr => inst_exp_ctr )
    port map ( a => inst_a, exp_offset => inst_exp_offset, no_detect => no_detect_inst,
ovfl => ovfl_inst, b => b_inst, exp_adj => exp_adj_inst );


end inst;
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW_norm_inst( inst_a, inst_exp_offset, no_detect_inst, ovfl_inst,
          b_inst, exp_adj_inst );

parameter a_width = 8;
parameter srch_wind = 8;
parameter exp_width = 4;
parameter exp_ctr = 0;


input [a_width-1 : 0] inst_a;
input [exp_width-1 : 0] inst_exp_offset;
output no_detect_inst;
output ovfl_inst;
output [a_width-1 : 0] b_inst;
output [exp_width-1 : 0] exp_adj_inst;

    // Instance of DW_norm
    DW_norm #(a_width, srch_wind, exp_width, exp_ctr)
      U1 ( .a(inst_a), .exp_offset(inst_exp_offset), .no_detect(no_detect_inst),
.ovfl(ovfl_inst), .b(b_inst), .exp_adj(exp_adj_inst) );

endmodule
```

## Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

| Date | Release | Updates |
|---|---|---|
| December 2020 | DWBB_202009.2 | ■ For STAR 3408988, corrected type for parameter exp_ctr in the example in "HDL Usage Through Component Instantiation - VHDL" on page 5 |
| July 2020 | DWBB_201912.5 | ■ Added the *exp_ctr* parameter to the examples on page 5 and page 6 |
| January 2019 | DWBB_201806.5 | ■ Updated example in "HDL Usage Through Component Instantiation - VHDL" on page 5<br>■ Added this Revision History table and the document links on this page |

# Copyright Notice and Proprietary Information



### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com